Point-Based Proxy and Interactive HDR Rendering From Unstructured Photographs

Boris Airieau, Daniel Meneveaux, Mohamed-Chaker Larabi

Point-Based Proxy and Interactive HDR Rendering From Unstructured Photographs

Boris Airieau, Daniel Meneveaux, Mohamed-Chaker Larabi

November 18, 2012

This paper presents a new image-based rendering method, based on photographs acquired with handheld cameras whose characteristics are unknown. Our system relies on recent calibration advances that use robust shape descriptors such as SIFT or SURF, associated with matching methods, producing a cloud of 3D points that belongs to the surface of acquired objects. The proposed method provides an interactive navigation system, with a high refresh rate, without any polygonal geometric reconstruction of objects. For a new viewpoint, our method performs the following operations: (i) point cloud projection onto the camera, (ii) depth interpolation in image space, (iii) reverse projection of input photographs to estimate the value of each pixel in the new view, (iv) occlusions management system for avoiding re-projection errors depending on the view. With this system, high dynamic range images can also be produced from freehand shooting.

1 Introduction

Recent advances in computer vision allow to ease the determination of intrinsic and extrinsic parameters of a camera from a set of freehand and unstructured captured photographs. This step is very important because it helps in finding the camera position and recovering depth information, that can be used for producing a 3D point cloud corresponding to the objects surface [FCSS10]. The entire processing framework is automated, and some authors have proposed interesting navigation systems based on tourist pictures of monuments [SSS06, SGSS08]. Some other approaches rather produce polygonal meshes [KSO04, KBH06, DLU11], optionally combined with textures.

Unfortunately, visualization methods used for such reconstructed environments do not fully benefit from the accuracy offered by actual photographs. On the one hand, point-based rendering systems allow realistic interactive rendering of objects without any polygonal mesh, but only use fixed (R, G, B) values associated with each 3D point. On the other hand, constructing meshes produces a high number of polygons, and requires the construction of dense textures whose resolution is fixed in advance, with often an impaired quality compared to original photographs.

Previous image-based rendering methods allow to use photographs for viewing objects or environments [AB91, LF94, GGSC96, BBM+01], but the acquisition systems are often complex, require some human intervention, and are difficult to manage for high dynamic range images (HDR) that require multiple shots per viewpoint [DM97].

This paper describes a new method for image-based interactive navigation from photographs taken freehand. Our goal is to propose a system fully exploiting the details offered by original photographs, with a high frame-rate, and avoiding mesh reconstruction. The proposed approach

takes advantage of methods such as *structure from motion* [HZ04, SSS06] for automatic calibration of photographs, followed by a point cloud densification [FCSS10]. Our visualization system has the advantage of being entirely on GPU, using only the point cloud, with the following contributions:

- construction of depth maps for each new view and original photographs, using a *pull-push* method;
- use these depth maps for determining the visible parts of objects;
- estimation of (R, G, B) pixel value for each viewpoint, determined only from relevant photographs;
- a model that handles epipolar constraints when using original photograph pixels for the observer viewpoint;
- a system for producing HDR images from multiple series of photographs taken freehand without any positioning constraint.

During visualization, for each new viewpoint, our method consists in projecting the set of 3D points on the virtual camera, constructing a depth map, and determining for each pixel the (R, G, B) value from original photographs. Occlusion issues are handled automatically by similarly constructing depth maps for each of the original viewpoints and using them as for shadow mapping algorithm. Our epipolar consistency model makes it possible to automatically adapt to resolution and orientation from the original photographs. The use of high dynamic range images allows the user to select the environment brightness during navigation. The obtained results show that a few tens of handheld camera images are sufficient for providing realistic rendering, with a framerate greater than 80 with a 512×512 image resolution.

The remainder of this paper is organized as follows: section two describes previous work on which our approach is based; section three presents the general architecture of our approach and tools used during pre-processing; section four describes our image-based rendering algorithm, the automatic management of occlusions and the creation of HDR images; section five presents and discusses our results; this paper ends with some conclusions and gives a number of openings in terms of future work.

2 Related work

Modeling geometric, photometric and radiometric properties of real objects is a tedious task, that requires both time and know-how. This is a reason why several authors have focussed on image-based methods, including visualization, modeling or re-lighting complex objects from photographs [DYB98, LH96, GGSC96, HED05, CEJ+06, SGSS08, DLD12].

Image-based navigation. Many methods allow visualizing objects from one or more photographs, using interpolation between viewpoints [CW93, LF94, SD96]. These methods often rely on the user's action for reconstruction, and the navigation part is performed with a small number of views to maintain the realism of photographs.

Adelson and Bergen introduced in 1991 the plenoptic function [AB91] to formalize the light energy distribution within an environment. This function represents the 5-dimensional luminance observed at each point and in each direction in space. A sampling of this function allows to produce new views, and the proposed implementation is based on a representation

of cylindrical images. *Lumigraphs* (or *light-fields*) [LH96, GGSC96] are also a representation of the plenoptic function using pairs of parallel planes, thus reducing the representation to a four-dimensional space instead of five. The acquisition of light-fields is nevertheless complex, requiring a large number of shots spread around the object. The method of concentric mosaics [SH99] offers a sampling of the plenoptic function using a representation based on concentric circles belonging to the same plane.

Some approaches additionally use a geometric representation (often called *proxy*) to reduce artifacts due to parallax errors [GGSC96]. For instance, Debevec et al. [DTM96, DYB98] defined textures depending on the viewpoint. Buehler et al. [BBM+01] propose an approach to generalize lumigraphs and view-dependent textures with the help of a mesh proxy, while Wood et al. use lumigraphs as textures [WAA+00]. Chaurasia et al. [CSD11] propose to use a point cloud as a proxy as well as the implementation of constraints on objects silhouettes to avoid distortions. For all these approaches, geometric models significantly reduce visual artifacts corresponding to parallax errors providing depth imperfections and photographs reprojection errors [PVGV+04]. However, the user intervention is required, and the processing and acquisition systems complexity make it difficult to use these methods for viewing environments from photographs. Recently, Davis et al. [DLD12] have also experimented various geometric representations, including object meshes, with a simplified acquisition system, based on interactive video acquisition, with camera control for providing a sufficiently precise object coverage. Our rendering method further simplifies the rendering process since only a few tens of unstructured photographs without coverage control still provide realistic results. In addition our system automatically handles unstructured photographs without any specific mesh reconstruction neither for the camera viewpoint nor for the object viewpoint.

In general, most methods of image-based navigation rely on advanced data acquisition systems requiring specific equipment. Some of them are based on the reconstruction of a geometric mesh [DTM96, DYB98, BBM+01] or require user intervention to guide the geometry reconstruction [CSD11]. Our goal is to provide an image-based visualization system without mesh reconstruction, where the user freely takes some photographs of the object, with as few constraints as possible, while preserving as many details as possible from photographs with a high refresh rate. This idea is also close to the concepts of unstructured lumigraphs / light fields [BBM+01, DLD12], without any reconstruction of geometric mesh and with per-pixel management of occlusions and photographs.

Multiview stereo. Our approach, similarly to other image-based rendering systems, operates on calibrated photographs, for whose intrinsic parameters (focal length) and extrinsic (position and orientation) are known. For this purpose, many methods exist in the literature with the use of patterns or dedicated data acquisition systems. The advent of efficient descriptors and moreover invariant to translation and rotation, such as SIFT [Low04] associated with systems of type *Structure from Motion* radically simplify the calibration of photographs. These systems coupled to matching methods such as RANSAC demonstrated their robustness on many image databases [SSS06, SGSS08]. Hartley and Zisserman [HZ04] present in detail the theory and algorithms used in stereo multi-view.

From photographs and calibration of cameras, *multi-view stereopsis* algorithms can be used to extract geometric information. Furukawa and Ponce propose a method [FP07] to generate a dense point cloud from a set of calibrated photographs and reconstruct triangular meshes. This method is then improved [FCSS10] by grouping photographs into clusters to perform parallel computations. Goesele et al. propose a method [GSC+07] based on the construction of a depth map for each of the photographs. These depth maps are then merged to represent the entire

viewed object, that can be used to generate a triangular mesh or a point cloud for visualization. We use the latter raw representation, which is very simple to manipulate and benefits from fast rendering with graphics processing units.

Point-based rendering. Visualization of point clouds has become a very important research topic largely explained by the availability of methods for acquiring objects using laser scanners. Many authors focused on the reconstruction of a geometric mesh [AB98,ACK01,KSO04,KBH06, DLU11], but these methods produce a multitude of triangles whose size is often no more than a pixel. This reduces dramatically visualization performances, and is subject to high memory consumption. In addition, point clouds are invariably noisy and the resulting triangulation edges are often unadapted to the reconstructed objects curvature.

To deal with these problems, some authors have proposed to directly render point clouds. Each 3D point is considered as an area associated with a position, a direction, an (R, G, B) value and a radius corresponding to the surface on which the point extends. The radius is generally based on the distance between the point and its nearest neighbors, so that surfaces associated with points overlap and avoid holes in the objects during rendering. QSplat method [RL00] organize points as a bounding spheres hierarchy where rendered primitives are oriented quadrilaterals which side length depends on the point radius. More recently, disks projection is performed on the GPU, directly on the display space with masks (or filters) depending on the disks orientation [ZPvBG01, RPZ02, ZRB+04].

To further enhance performances, some authors propose to work exclusively in image space after points projection. A pyramid of depth maps is used to interpolate depths between points. This approach, called *pull-push* is originally inspired by [GGSC96]. Grossman and Dally [GD98] perform reconstruction of depth maps from dense point clouds and Pfister et al. [PZvBG00] fill gaps between surface elements. Recent approaches suited to graphic cards offer outstanding performance [MKC07].

Rosenthal and Linsen propose another approach based on an efficient point-based rendering avoiding *pull-push* [RL08], but we chose the implementation proposed in Marroquim et al. [MKC07] because surface element orientation is handled more accurately, and remains independent of point clouds density.

3 Work Overview

Figure 1 shows the software architecture of our visualization system. Photographs are used as input of a pre-processing step that automatically determines intrinsic and extrinsic parameters of each camera viewpoint, as well as a set of 3D points corresponding to the object surface. The resulting point cloud is used for reconstructing depth for both existing photographs and new viewpoints (geometric proxy). Calibrated photographs are directly employed for visualization. With several series of photographs at varying exposure, our system can automatically manage the construction of HDR images.

Photographs are calibrated using *structure from motion*. For each photograph C_i a set of SIFT descriptors [Low04] is determined; matching is then performed using a kd-tree data structure and a nearest neighbors approach. An adjustment algorithm (*bundle adjustment*) following the RANSAC method [Sna08] allows to estimate intrinsic parameters (focal length and lens distortion factors) and extrinsic parameters (camera position and orientation) for each camera viewpoint C_i, providing a projection matrix M_i. The second pre-processing step intends to produce a dense point cloud of the objects surface (*multi-view stereopsis*) [FCSS10] associated with a (R, G, B) color and an orientation. Each 3D point P_j can thus be projected onto an

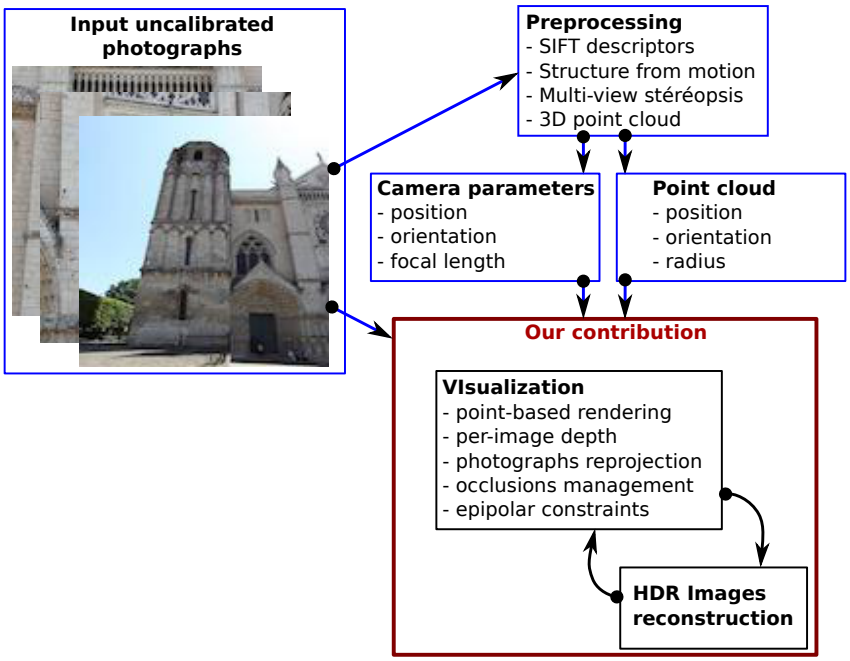


Figure 1: Software architecture of the developed system of image-based rendering. Input images are used to determine intrinsic and extrinsic camera parameters as well as the point cloud. Our system makes use of both information for HDR image reconstruction and interactive visualization.

image C_i in P_j^i as follows: $P_j^i = M_i P_j$. This allows in particular to define the depth associated with each pixel located at P_j^i (i.e. in this case, the original photographs).

A radius r_j is associated with P_j, according to the neighborhood of 3D points, so that P_j surface elements overlap. To actually calculate this information, we use a *k-nearest neighbors* algorithm and a *Kd-tree* data structure.

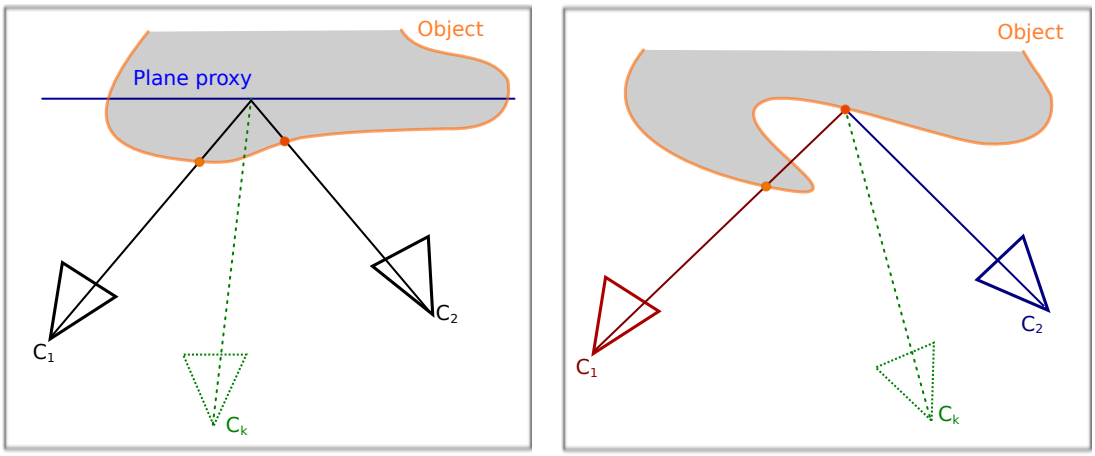
During visualization, for each observer viewpoint, all points P_j are projected onto image space, and we use the algorithm proposed by Marroquim *et al.* [MKC07] to define a depth at each pixel. The (R, G, B) color of each pixel of the image is estimated by inverse projection on each photograph, taking occlusions into account, and weighting the viewing direction of each active camera. The same process is applied for HDR image reconstruction [DM97], given several sets of photographs with varying exposure.

4 System modeling

The input parameters of our rendering system comes from the calibration process: a point cloud used as a geometric proxy and calibrated images associated with their respective projection matrices. Figure 2 illustrates the point cloud for an object as well as the photographs position, for a real object. For a new rendering viewpoint C_k, defined by a projection matrix M_k, our system consists in: (i) reconstructing the object depth per pixel; (ii) estimating the 3D corresponding position; (iii) determining the (R, G, B) color with a back projection on the original photographs. However, this last step should be carefully managed because of the unstructured data [BBM+01].

Figure 2: Calibration results: point cloud and calibrated photograph viewpoint images.

Geometric proxy. Parallax errors can be efficiently reduced with geometric proxies [GGSC96, BBM+01, PVGV+04]. The goal is to determine the object depth observed through each pixel of the reconstructed image in order to precisely identify the corresponding pixels on the original photographs. Figure 3 illustrates two cases that demonstrate the importance of depth management, the artifacts corresponding to these errors are ghosting or exaggerated smoothing. Our method relies on a point-based proxy, with a per-pixel depth reconstruction for image C_k (described in section 5.1).



(a) Unsuitable plane proxy

(b) Occlusion management

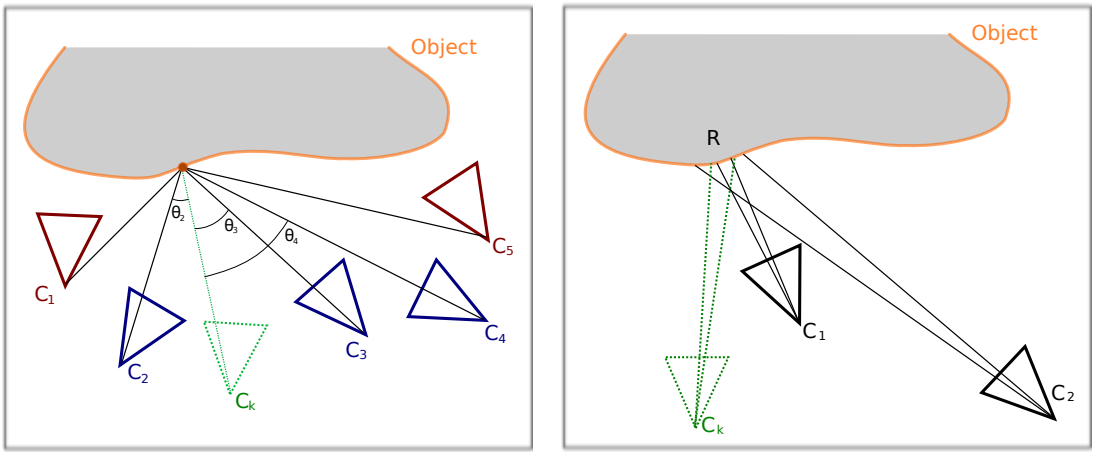
Figure 3: Choice of geometric proxy. (a) With a plane proxy, the resulting (R, G, B) color on camera C_k is not correctly estimated; (b) Camera C_1 should not be taken into account since the observed region is not the same as C_k for the considered pixel. In this case, ghosting would appear on C_k.

Epipolar consistency. Many reconstruction systems fail with glossy objects due to high frequency radiance variations corresponding to mirror effects, very sensitive to the observer position. Still, they remain robust for non perfectly diffuse objects and the angular deviation between C_k and C_i images should be taken into account during rendering, to preserve as many

visual details as possible (Fig. 4.a). In addition, for a new viewpoint C_k, observing directions are aligned with a camera C_i center of projection, the (R, G, B) value should be the same as the corresponding C_i pixels. Our approach makes use of all the photographs that can contribute to each pixel of C_k. This accounts for reducing angular deviation artifacts and avoiding flickering effects when the viewpoint changes. Instead of using penalties (as described in [BBM+01]), we use significance weights associated with photographs for each pixel of a new image C_k:

$$I_k(x,y) = \frac{1}{\sum_{i=1}^{N} cos\ \theta_i} \sum_{i=1}^{N} I_i(x',y') \times cos\ \theta_i,$$

with $I_k(x,y)$ being the current C_k pixel, $I_i(x',y')$ the corresponding pixel on image i and θ_i the observation angle. These weights account for non diffuse objects, with slight glossy effects, for instance.



(a) Orientation variations (b) Resolution variations

Figure 4: (a) Angular deviation between C_k and C_i: observation direction should be taken into account; (b) relative camera positions also introduce resolution differences so that observing distances should also be considered.

Resolution. Another important parameter is the image resolution, since the observed details are related to the distance between the object and the camera. For each image pixel, the (R, G, B) value corresponds to the integral of reflected radiances on the object surface, toward the camera. Figure 4.b illustrates the observation of a same region $\mathcal{R}$ with two photographs C_i^1 and C_i^2 through two respective pixels I_1 and I_2. The solid angle corresponding to pixels do not cover perfectly the same surface on the object [BBM+01]. Again, this difference should be taken into account, and we introduce the following weighting function:

$$I_k^r(x,y) = \left(\sum_{i=1}^{N} \delta(C_k, P_{x,y}, C_i) + 1 \right) \sum_{i=1}^{N} \frac{I_i(x',y')}{\delta(C_k, P_{x,y}, C_i) + 1},$$

where P_{xy} corresponds to the observed surface point, $\delta(C_k, P_{x,y}, C_i) = abs(||C_k - P_{x,y}|| - ||C_i - P_{x,y}||)$ is the distance difference between $P_{x,y}$ and the centers of projection of C_i and C_k.

HDR images. Constructing high dynamic range images [DM97] requires either several LDR (Low Dynamic Range) photographs from the same viewpoint with varying exposure or a specific

hardware. Our system offers an automatic method for producing such images based on several series of handheld cameras, each series having a fixed exposure, without any need to produce several exposure with the same viewpoint. The principle is illustrated on Figure 5. For a new viewpoint C_k, an image is produced with our rendering system according to each series independently, providing for this viewpoint a set of LDR images, merged to produce the final HDR image. This reconstruction can be performed either during interactive visualization, or as a pre-computation aiming at reconstructing a unique series of HDR images as input of our rendering system. We have chosen the second solution for two main reasons: (i) the number of images increases linearly with the number of chosen exposition values, increasing the needs for memory on the GPU; (ii) the time required for merging these images cannot be neglected. Another issue is linked to the choice of these viewpoints. Since each of the image series should cover the object surface, we have chosen to pick the viewpoint of one series, and every LDR image is replaced by a reconstructed HDR image (with $\{C_k\} = \{C_i\}$). The other series thus become useless during interactive visualization.

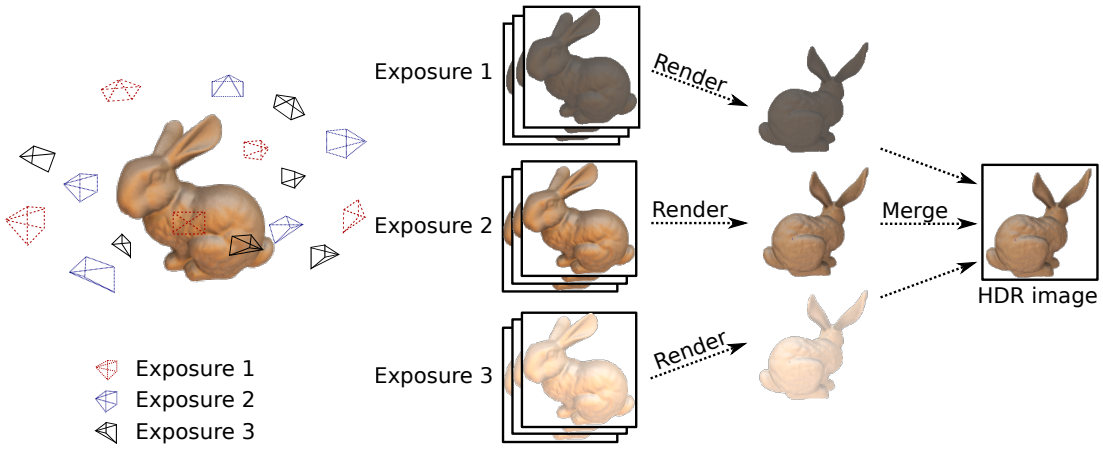


Figure 5: HDR image reconstruction from three series of photographs.

Our rendering system is completely ran on the GPU, so that the CPU only has deals with GUI events and can be also employed for other tasks during visualization. Note that HDR images can also be constructed on the GPU in a few seconds.

5 Implementation

The proposed image-based rendering system implements all the above models and mechanisms. For each new viewpoint C_k, the point-cloud is used for constructing a depth map. Photographs are used to set per-pixel (R, G, B) values. Let us recall that each point cloud vertex is associated with: a 3D position P_j, a normal n_j, and a radius r_j.

5.1 Depth reconstruction

Given the M_k matrix corresponding to the C_k camera, each point cloud vertex P_j is projected onto the screen plane. Each point depth is stored as well as its normal vector N_j, and its radius r_j into two textures T_{zn} and T_{rc}. T_{zn} stores depth z_j and the vertex normal $\vec{n}_j$ while T_{rc} stores the radius r_j and the coordinates of the corresponding projected disk in the image plane (thus defined by an ellipse). From this incomplete 2D representation, the goal is to recover the depth for all image pixels. Our method corresponds to the approach proposed in [MKC07], where a

depth image pyramid is constructed (Figure 6) and a *pull-push* algorithm fills the inter-pixel space.

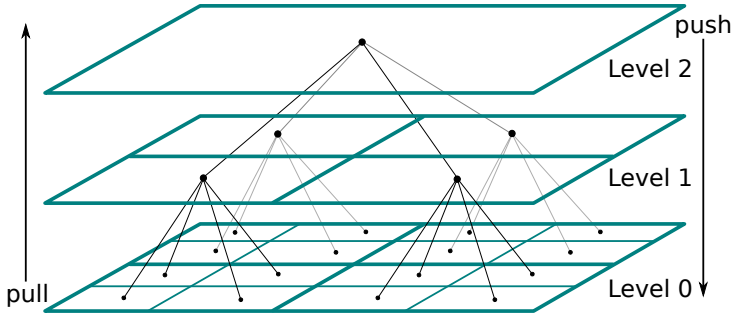


Figure 6: Images pyramid and pull-push

Pull stage. The goal of the pull stage is to use the most detailed images in the pyramid, to fill progressively the upper levels. A given depth is associated with each pixel to eliminate occluded parts. At the lowest level, the lower bound is the point depth and the upper bound is the point depth plus the point radius (r_i). For each level, the pixel intervals are set to the minimum and maximum depth of the lower level corresponding pixels. A pixel is considered as occluded if its depth does not belong to the smallest screen depth interval. The region of influence of a pixel is delimited by an ellipse corresponding to the projected disk surface element. However, the ellipse center has to be maintained relatively to the pyramid image level (see Figure 7.c). Therefore, a displacement vector is used to maintain the necessary precision. Ellipses parameters are actually used during the *push* stage.

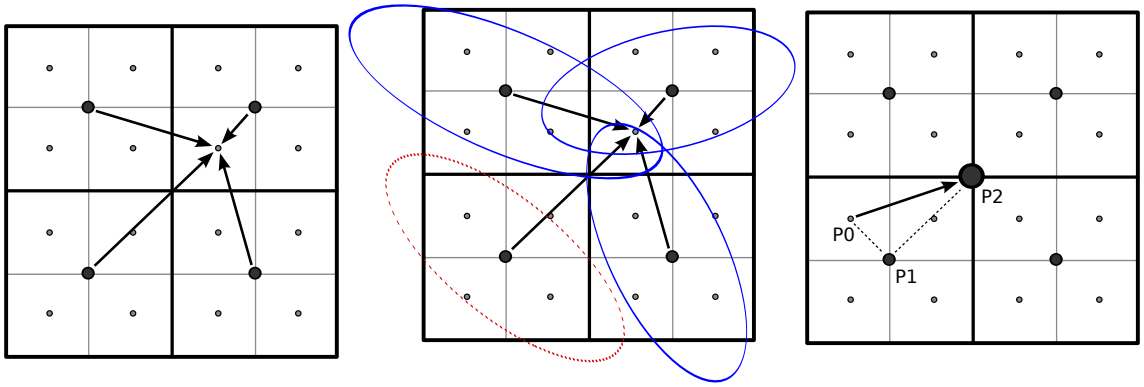
Push stage. The goal is to propagate the depth information from the root image through the whole pyramid. Therefore, data are propagated for one (lower level) pixel from the four corresponding upper level pixels (Fig. 7). For each of these pixels, the ellipse determines whether the upper pixels parameters should be taken into account or not. Figure 7.c illustrates an example where only three over four pixels are used to construct the lower level parameters. All unprocessed pixels at the lowest level are set to a depth equal to zero.

5.2 Occlusion management

The above per-image depth reconstruction process can be applied to each C_k camera image, providing a 3D point P'_k corresponding to the object surface at each pixel. This process can naturally be applied to the original photographs C_i, so as to estimate a depth map for each of them. Our goal is to solve the problems illustrated on Figure 3. C_k depth maps are used as shadow maps [Wil78] so that depth corresponding to each point P'_k projected onto C_i images can be compared. When both depth match, the C_i (R, G, B) pixel value can be taken into account; otherwise, the pixel does not correspond to the same region of the object, it should thus be rejected. Figure 8 illustrates the gain brought by occlusion management with one of our scenes.

5.3 Rendering

For a given C_k viewpoint, depth reconstruction and occlusions management is performed for each pixel, providing the set of valid C_i images pixels. The final (R, G, B) color is obtained



(a) (b) (c)

Figure 7: Ellipses parameters management during the pull-push stage. Upper level pixels (in bold) are used to estimated the parameters of the next lower level. In some cases, pixel ellipses do not cover the lower level pixels.

Figure 8: Occlusions management for a new viewpoint C_k. Left: for each pixel of C_k, C_i images are used only when depth corresponds. Right: all images are used for producing the C_k image, and ghosting effects appear.

with a combination of these pixels. Integrating the model described in Section 4 leads to the following equation:

$$I_k(x,y) = \frac{1}{\sum_{i=1}^{N} \frac{cos\ \theta_i}{\delta(C_k,P_{xy},C_i)}} \sum_{i=1}^{N} \frac{I_i(x',y')cos\theta_i}{\delta(C_k,P_{xy},C_i)}.$$

Figure 9 shows the results obtained with our method compared with usual point-based rendering approaches where only vertices properties are taken into account.

Since all used photograph pixel colors remain close in practice, we have used the euclidean distance in the RGB color space. We did not see any false colors in our rendering system. However, it would be possible to use another color space, closer from human perception if needed.

When several series of images (each of them with its own exposure level) are available, the above process is performed independently for each series and for a given C_k viewpoint. An HDR image is eventually constructed from the resulting set of rendered images. Figure 10 shows several image series, with varying exposure levels while Figure 11 presents tone-mapped

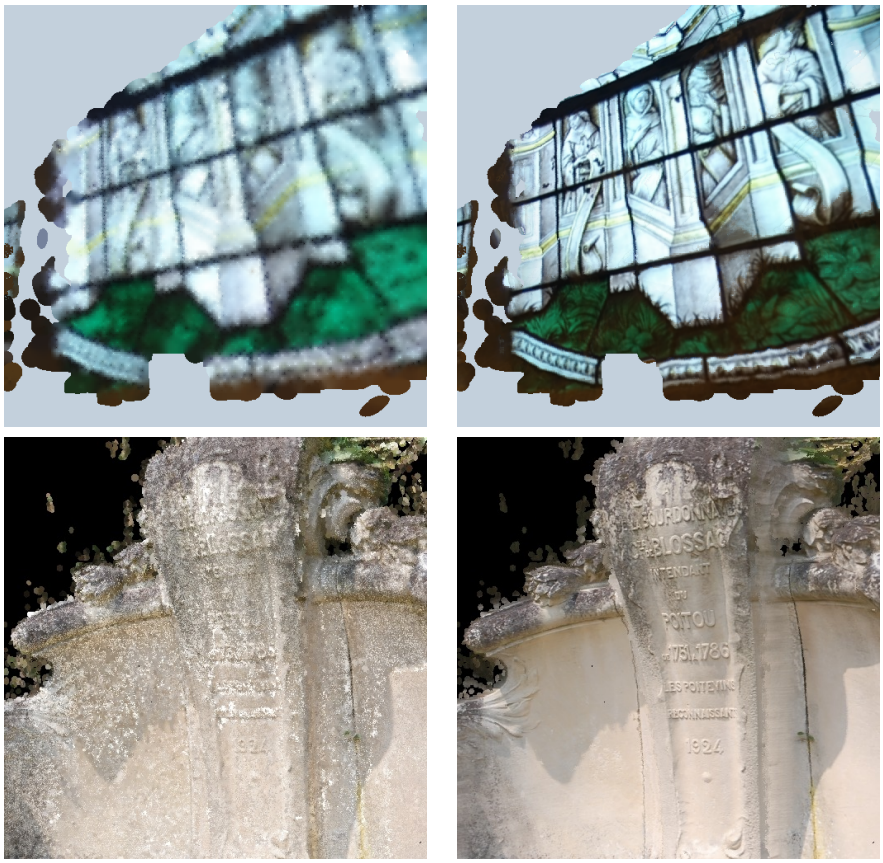


Figure 9: Point-based rendering (left); our image-based rendering approach (right).

HDR images produced by our rendering system.

6 Results

The computer used for providing the results of this paper is equipped with an Intel Core I7 920 processor as well as a NVIDIA 570 GTX graphic processing unit. The program has been implemented in C++, with OpenGL library and GL shading language for programmable graphics hardware shaders. The original photographs resolutions vary from 1500×1000 to 5600×3700 pixels, and their storage on the GPU uses OpenGL texture compression. All the reconstruction process is performed on the CPU, while our image-based rendering program is fully achieved on the graphics processing GPU.

Table 1 and Figure 12 present the five test scenes used in this paper with: the number of 3D vertices provided by the pre-computation system, the number of original photographs, and the memory requirements during interactive visualization.

Table 2 indicates running time corresponding to each task, as well as the frame-rate obtained during visualization, with two image resolutions: 512×512 and 1024×1024 pixels. *Rasterization* column corresponds to the point cloud projection time; *Pullpush* corresponds to depth reconstruction; and *Accum* indicates photographs pixels back-projection. This table shows that performance rather varies according to rasterization (i.e. the number of 3D points used as geometric proxy) and texture requests, while depth reconstruction time remains constant, though

Figure 10: Three series (rows) samples with various exposures. Each row presents two of the photographs we acquired with one given exposure level.

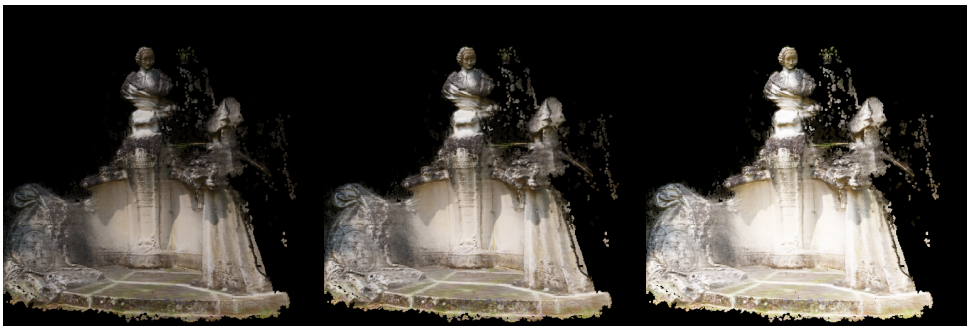


Figure 11: HDR images with varying tone-mapping parameters applied to our reconstructed HDR images.

Table 1: Scenes characteristics, with required memory.

Scene	# points	# img	Σ mem.	Vertices	Textures
Cathedral	3799742	58	743 Mo	160 Mo	340 Mo
Church	1172590	20	456 Mo	50 Mo	279 Mo
Sculpture	298515	62	192 Mo	13 Mo	92 Mo
Statue	3819846	100	574 Mo	161 Mo	172 Mo
Bell	517466	236	330 Mo	20 Mo	196 Mo

depending on image resolution.

Figure 13 shows running time according to the number of used input photographs. The X-axis scale is not the same for all scenes, since we have chosen to illustrate several cases

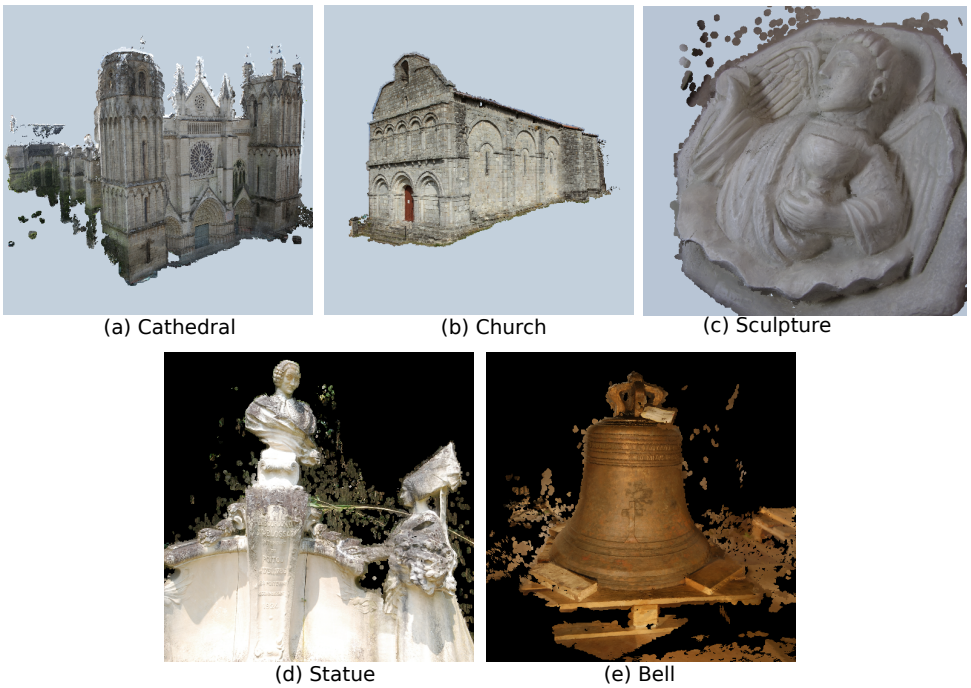


Figure 12: Five test scenes used in this paper, displayed with our interactive image-based rendering system.

Table 2: Running time for 512×512 / 1024×1024 image resolutions.

Scene	Raster. (ms)	Pullpush (ms)	Accum (ms)	Total (ms)	Fps
Cathedral	7.4 / 7.6	1.3 / 3.4	2.7 / 7.0	11.8 / 18.5	85 / 54
Church	2.7 / 2.8	1.3 / 3.4	1.3 / 3.3	5.7 / 10.3	175 / 97
Sculpture	0.7 / 0.8	1.3 / 3.4	2.3 / 8.2	4.6 / 13.0	215 / 77
Statue	7.5 / 8.5	1.3 / 3.4	2.8 / 10.3	11.9 / 22.7	84 / 44
Bell	1.0 / 1.1	1.3 / 3.4	8.9 / 25.5	11.6 / 30.7	86 / 33

with various number of images. In practice, running time remains near linear according to the number of used photographs (until 200).

We have also compared our images with a mesh reconstruction associated with textures (123D catch proposed by Autodesk [Aut]). The reconstruction process could be performed for three of our test scenes: Church (304 928 triangles), Sculpture (106 897 triangles) and Bell (400 000 triangles); it failed for the others. With this tool, the atlas textures resolution is 4096 × 4096 pixels. The Sculpture scene mesh is very precise and texture mapping provides realistic images. It represents the best result we could obtain with mesh reconstruction. In other cases, the triangulation and texture mapping is not adapted to geometric details, as shown on Figure 14 with the Church scene.

The proposed method has several advantages compared to textured triangular meshes. First, visual details are automatically adapted according to the viewpoint, taking into account progressively levels of details, and directional information, thanks to the weighting function we introduced. Presently our method requires to store all the images in the GPU, potentially with high memory consumption. Reducing photographs resolution can be a solution in some cases (see Figure 15), though reducing image quality. Loading photographs on demand according to

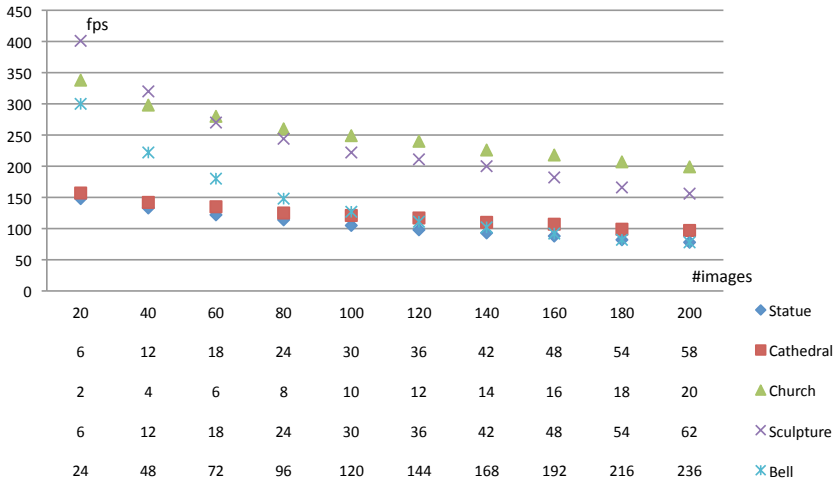


Figure 13: Frame-rate according to the number of used input photographs, rendered with 512×512 resolution.

(d) Our method (e) 123D Catch (f) 123D Catch - mesh

Figure 14: Comparison between our approach and textured meshes. Our image-based rendering system (left), 123D catch from Autodesk (middle). produced polygonal mesh (right).

the viewpoint could be another option. However, the latter would necessarily act upon performances due to bandwidth limitations. In addition, the results show that only a few tens of photographs provide enough data for our image-based rendering system, offering good frame rates with up to two hundred images.

In some cases, the lack of photographs of some parts of the object leads to a decrease in

(a) 5616×3744 pixels (original), 456 Mo (b) 1/4 size, 297 Mo (c) 1/16 size, 257 Mo

Figure 15: Church visualization with various resolutions of input photographs, with corresponding memory usage.

terms of 3D points density produced by the calibration system and creates thus holes on the object surface. Our current research aim at detecting these holes on images and interpolate depth for adding new 3D points. However this problem should be carefully considered since holes also depend the object topology (for instance the handle of a cup should not be filled).

7 Conclusion and future work

This paper describes a new image-based rendering method allowing interactive visualization within environments acquired from handheld cameras. A pre-computation stage consists in calibrating all viewpoints from the input photographs and construct a point cloud corresponding to the object surface, used as a geometric proxy. Our system is entirely performed on GPU and keeps the CPU free.

Depth management and epipolar consistency are managed according to the user viewpoint during interactive visualization, so that each photograph pixel be properly taken into account for each new viewpoint. This management makes our method auto-adaptive in terms of image resolution and orientation. Our system also allows to construct HDR images, from several series of photographs, each of them having a fixed exposure.

Our approach does not require any user intervention during the reconstruction process contrary to most image-based acquisition systems. It offers the advantages of image-based visualization systems while avoiding the usual problems of mesh reconstruction for geometric proxies. Only few tens of unstructured photographs are sufficient for providing interactive and realistic visualization.

We currently aim at defining topological 2D configurations for filling holes provided by point based rendering, and eventually add new 3D points to the cloud. A longer range term research concerns the use of environment illumination, so as to interactively combine image-based rendering and relighting with the same series of photographs. HDR light probes might provide a good starting point to this study, BRDF parameters could be recovered, provided that photographs distribution be more dense. We believe that the system proposed by Davis et al. in [DLD12] could actually guide photographs distribution.

8 Aknowledgements

This work has been funded by the FEDER NavII project (Poitou-Charentes region, France). The authors also wish to thank Thomas Vieweger for the plaster statue.

References

[AB91] Edward H. Adelson and James R. Bergen. The plenoptic function and the elements of early vision. *Computational Models of Visual Processing*, pages 3–20, 1991.

[AB98] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. In *Proceedings of the fourteenth annual symposium on Computational geometry*, SCG '98, pages 39–48, New York, NY, USA, 1998. ACM.

[ACK01] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust. In *Proceedings of the sixth ACM symposium on Solid modeling and applications*, SMA '01, pages 249–266, New York, NY, USA, 2001. ACM.

[Aut] Autodesk. 123d catch.

[BBM+01] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pages 425–432, New York, NY, USA, 2001. ACM.

[CEJ+06] Charles-Félix Chabert, Per Einarsson, Andrew Jones, Bruce Lamond, Wan-Chun Ma, Sebastian Sylwan, Tim Hawkins, and Paul Debevec. Relighting human locomotion with flowed reflectance fields. In *Annual Conference on Computer Graphics*, 2006.

[CSD11] Gaurav Chaurasia, Olga Sorkine, and George Drettakis. Silhouette-aware warping for image-based rendering. *Computer Graphics Forum*, 30(4), 2011.

[CW93] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 279–288, New York, NY, USA, 1993. ACM.

[DLD12] Abe Davis, Marc Levoy, and Fredo Durand. Unstructured light fields. *Comp. Graph. Forum*, 31(2pt1):305–314, May 2012.

[DLU11] Klaus Denker, Burkhard Lehner, and Georg Umlauf. Real-time triangulation of point streams. *Engineering With Computers*, 27:67–80, 2011.

[DM97] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Annual Conference on Computer Graphics*, pages 369–378, 1997.

[DTM96] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *Proc. of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 11–20, New York, NY, USA, 1996. ACM.

[DYB98] Paul E. Debevec, Yizhou Yu, and George Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In *Eurographics Symposium on Rendering/Eurographics Workshop on Rendering Techniques*, pages 105–116, 1998.

[FCSS10] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Towards internet-scale multi-view stereo. In *Computer Vision and Pattern Recognition*, pages 1434–1441, 2010.

[FP07] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multi-view stereopsis. In *Computer Vision and Pattern Recognition*, 2007.

[GD98] J. P. Grossman and William J. Dally. Point Sample Rendering. In *Eurographics Symposium on Rendering/Eurographics Workshop on Rendering Techniques*, pages 181–192, 1998.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Annual Conference on Computer Graphics*, pages 43–54, 1996.

[GSC+07] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. Multi-view stereo for community photo collections. In *International Conference on Computer Vision*, pages 1–8, 2007.

[HED05] Tim Hawkins, Per Einarsson, and Paul E. Debevec. A dual light stage. In *Eurographics Symposium on Rendering/Eurographics Workshop on Rendering Techniques*, pages 91–98, 2005.

[HZ04] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[KBH06] Michael M. Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Symposium on Geometry Processing*, pages 61–70, 2006.

[KSO04] Ravikrishna Kolluri, Jonathan Richard Shewchuk, and James F. O'Brien. Spectral surface reconstruction from noisy point clouds. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '04, pages 11–21, New York, NY, USA, 2004. ACM.

[LF94] S. Laveau and O. D. Faugeras. 3-d scene representation as a collection of images. In *Twelfth International Conference on Pattern Recognition (ICPR'94)*, volume 1, pages 689–691, 1994.

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 31–42, New York, NY, USA, 1996. ACM.

[Low04] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[MKC07] Ricardo Marroquim, Martin Kraus, and Paulo Roma Cavalcanti. Efficient point-based rendering using image reconstruction. In *Proceedings Symposium on Point-Based Graphics*, pages 101–108, 2007.

[PVGV+04] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *Int. J. Comput. Vision*, 59(3):207–232, September 2004.

[PZvBG00] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus H. Gross. Surfels: surface elements as rendering primitives. In *Annual Conf. on Computer Graphics*, pages 335–342, 2000.

[RL00] Szymon Rusinkiewicz and Marc Levoy. Qsplat: a multiresolution point rendering system for large meshes. In *Annual Conference on Computer Graphics*, pages 343–352, 2000.

[RL08] Paul Rosenthal and Lars Linsen. Image-space point cloud rendering. In D. Thalmann T. Capin, T.-S. Chua, editor, *Computer Graphics International*, pages 136–143, 2008.

[RPZ02] Liu Ren, Hanspeter Pfister, and Matthias Zwicker. Object space EWA surface splatting: A hardware accelerated approach to high quality point rendering. In *Computer Graphics Forum (Eurographics 2002)*, volume 21, pages 461–470, 2002.

[SD96] Steven M. Seitz and Charles R. Dyer. View morphing: synthesizing 3d metamorphoses using image transforms. In *Annual Conference on Computer Graphics*, 1996.

[SGSS08] Noah Snavely, Rahul Garg, Steven M. Seitz, and Richard Szeliski. Finding paths through the world's photos. *ACM Transactions on Graphics (SIGGRAPH 2008)*, 27(3):11–21, 2008.

[SH99] H. Y Shum and Li-Wei He. Rendering with concentric mosaics. In *Annual Conference on Computer Graphics*, pages 299–306, 1999.

[Sna08] Noah Snavely. Bundler, 2008. http://phototour.cs.washington.edu/bundler/.

[SSS06] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3d. In *SIGGRAPH Conference Proceedings*, pages 835–846, New York, NY, USA, 2006. ACM Press.

[WAA+00] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface light fields for 3d photography. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 287–296, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[Wil78] Lance R. Williams. Casting curved shadows on curved surfaces. *ACM Siggraph Computer Graphics*, 12:270–274, 1978.

[ZPvBG01] Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface splatting. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 371–378, New York, NY, USA, 2001. ACM.

[ZRB+04] Matthias Zwicker, Jussi Räsänen, Mario Botsch, Carsten Dachsbacher, and Mark Pauly. Perspective accurate splatting. In *Graphics Interface*, pages 247–254, 2004.